Heavy Changes in the Input Flow for Learning Geography of a Robot Environment

Georgii Khachaturov(✉), Josué Figueroa-González,
Silvia B. González-Brambila, and Juan M. Martínez-Hernández

Metropolitan Autonomous University - Azcapotzalco,
San Pablo Av. 180, Reynosa Tamaulipas, 02200 Mexico City, Mexico
{xgeorge,jfgo,sgb}@correo.azc.uam.mx, martinez.juan.hdz@gmail.com

Abstract. A novel approach to generation of geographic knowledge from robot views is presented. It is implemented in a pilot software where a virtual robot operates in a static 2D-environment. The robot sensor scans with rays an angular field of view and produces a 1D view of distances to the closest obstacle. By processing such views, 'heavy changes' are detected to trigger switching local maps in an atlas that represents geography of the robot environment. To detect heavy changes, firstly, each plot is transformed to a string of singular points; then, in time-scale, a pair of such strings is subjected to a treatment based on application of the distance of Levenshtein, which leads to so-called Editorial Prescription (EP); a heavy change is detected if EP shows a considerable distinction between strings. This approach is applied in automatic construction of an atlas for non-Cartesian navigation, while robot explores the scene.

Keywords: Detection of heavy changes · Levenshtein distance · Map switching · Non-Cartesian navigation · Processing range images

1 Introduction

The particular problem treated in this paper is related to a wider research: the one of non-Cartesian robot navigation. Its specific is illustrated below with some views generated by a pilot software that simulates evolution of a virtual robot in an environment, guided by supervisor. The environment is static, bi-dimensional, and populated with obstacles. Figure 1a shows an instance of such environment; it is a supervisor's view of a scene; the robot itself is shown as a dark triangle at the bottom. If the robot would be supplied with a color camera, it could see the same scene as Fig. 1b shows. However, it has a different kind of sensor: the one that scans all directions in a field of view and measures a distance to the first obstacle along each direction; each distance is subjected to a transformation so that the final view obtained by the robot for the same scene is the one shown in Fig. 1c. Note that views like Fig. 1c are called below '*angle-distance plots*', but this should not to confuse the readers: they show not crude, but somehow transformed distances.

L. Rutkowski et al. (Eds.): ICAISC 2017, Part I, LNAI 10245, pp. 518–529, 2017.
DOI: 10.1007/978-3-319-59063-9_46

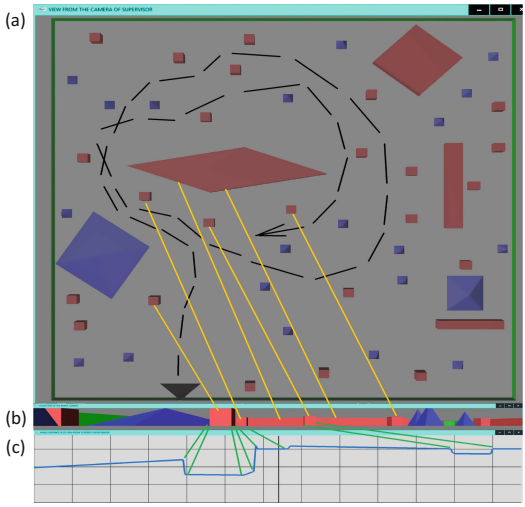

Fig. 1. Three views for the same "robot-in-environment" scene: **(a)** A supervisor's view; the robot position and orientation is shown by a dark triangle at the bottom; the dashed line is commented in Sect. 5. **(b)** A robocentric view by a color camera. **(c)** A robocentric view by an 'angle-distance plot'. The lines between (a) and (b), and between (b) and (c) connect the same objects and the same details shown in the views. (Color figure online)

A single view like the one in Fig. 1a provides the supervisor with full information about geography of the environment. But it is unavailable to robot, so we try to help the robot to reach an equivalent understanding of the geography, but by processing a set of views similar to that in Fig. 1c.

The process of learning geography is based on driving the robot along a continuous trajectory in the scene, combined with interpretation of the input flow of robot views. The process starts from an initial robot position in the scene, like the one of Fig. 1a.

This work presents an approach to a particular issue of processing the input flow: the detection of '*heavy changes*' or, in other words, those changes in the flow, which can be used as milestones for the non-Cartesian navigation. Note that robot should run a permanent process of self-localization for identifying the map in the atlas to which the robot's current state belongs (so called *current map*). While no heavy change occurs, the robot stays in the same current map; otherwise it must switch the current map to other map of the atlas or, if no appropriate map exists, firstly to introduce a new map into the atlas and only then switch the current map to it. That is, the heavy changes are the main interface between the robot sensor system and an inner representation of geography.

Our approach is implemented in the scope of the pilot software mentioned above and its principles are verified by experiments with this software.

The rest of paper is organized as follows: Sect. 2 offers a brief review of previous works and some concepts used in this paper. Section 3 describes a

processing scheme for detection of heavy changes. Section 4 develops furthermore the draft of Sect. 3, namely it deals with specific issues of the Levenshtein distance. Section 5 describes the pilot software and experiments on it. Section 6 contains a conclusion.

2 Used Concepts and Related Works

2.1 The Levenshtein Distance

The Levenshtein distance [1] is a measure of the difference between two strings and it can be described as the minimum number of operations over single letters needed to convert or change one word into another. Such a step-by-step conversion is called *editorial prescription*. The process for obtaining this measure was developed by Vladimir Levenshtein in 1965. In general case it depends on three functions: *w(p, q)* – the cost of replacement of symbol p with symbol q, *w(ε, q)* – the cost of insertion of symbol q, and *w(p,ε)* – the cost of deletion of symbol p. Let string a should be converted to string b, then the following formula (1) determines recursively all elements of a rectangle matrix

$$D_{a,b}(i,j) = \begin{cases} 0 \text{ if } j = i = 0; \text{ else :} \\ D_{a,b}(i-1,0) + w(\varepsilon, a_i) \;||\; D_{a,b}(0, j-1) + w(b_j, \varepsilon) \\ \text{for } j = 0 \;||\; i = 0; \text{ otherwise :} \\ min \begin{cases} D_{a,b}(i-1, j) + w(\varepsilon, b_j) \\ D_{a,b}(i, j-1) + w(a_i, \varepsilon) \\ D_{a,b}(i-1, j-1) + w(a_i b_j) * 1_{(a_i \neq b_j)}, \end{cases} \end{cases} \tag{1}$$

where i and $j \geq 0$ run indexes of symbols in respective strings, $1_{(a_i \neq b_j)}$ is so called *indicator function* equal to 0 when $a_i = b_j$ and to 1 otherwise. In particular case, when *w(ε, q)* = 1, *w(p, ε)* = 1, *w(p, p)* = 0, and *w(p, q)* = 1 as p ≠ q, this matrix defines the Levenshtein distance between a and b. The Levenshtein distance is widely known thanks to the applications used in detecting plagiarism in text or codes [2], and recently in dialect analysis [3]; some uses of this technique related with image processing can be found in [4]. Here it is applied to the strings constructed as a coded form of the angle-distance plot. Such strings are quite short so we do not meet difficulties specific for the detecting plagiarism in large texts.

2.2 Non-Cartesian Navigation

Navigation for humans and animals does not assume pointing the goal as a point in a Cartesian space. The attempts to understand how the human brain tackles the navigation tasks have been undertaken in numerous works, during several decades mainly in psychology. The first mathematically strict model of non-Cartesian navigation was proposed by Khachaturov [5]. Other works on this topic are [6–8].

The model presented in [5] (so called GT-model) is formed by two graphs: G which stands for the robot sensor-motor knowledge, and T represents a geography or administrative system imposed on a corresponding environment. In formal terms, a *GT-model* is defined as pair $\{G, T\}$, where the components satisfy the following properties:

- $G = (V_G, E_G)$ is a directed graph with non-negative weights assigned to its edges;
- $T = (V_T, E_T)$ is a directed graph with an unique source-node whereas the set of its terminal nodes coincides with V_G.

Interpretation of nodes of V_G and edges of E_G follows. Let S denote the space of all robot states inside an environment. It is assumed that *robot views* are put in one-to-one correspondence with points of S. A set of close in a metric robot views generates a neighborhood of close states in S. Each node of V_G stands for a domain of S and the whole set of nodes V_G represents a sampling of S generated by equivalent in a certain meaning robot views.

An edge $l \in E_G$, where $l = (a,b)$ and $a, b \in V_G$, represents a control rule that drives robot to change any view associated with a to a view associated with b. Graph T represents a system of sets composed by nodes of G. This system is determined by the following rule: *Any node* α *of* V_T *corresponds to set* $V_\alpha \subset V_G$ *defined as the set of all terminal nodes of all paths in* T *that start from* α. Intuitively, α is the name of a geographic object represented by the set V_α – *the domain of* α. An edge of E_T from node β to node α stands for inclusion of respective domains, $V_\beta \supset V_\alpha$, so the first geographic object of an edge is wider than the second.

Since a terminal node of T corresponds to a node of V_G, it represents some close states of a robot inside its environment. Unlike that, for a non-terminal $\alpha \in V_T$, its domain V_α can contain far robot states. Graph T organizes such domains into a system quite similar to a real system of geographical concepts. In particular, the domain of the source-node of V_T is the whole V_G. If there is a path in T from α to β, then the domain of α obviously contains the domain of β. The simplest kind for T is a tree. If T is a tree, it represents so-called *tree decomposition* of G, [9].

In these terms, formalization of a non-Cartesian navigation problem is as follows. Let $\alpha, \beta \in V_T$ and V_α, $V_\beta \subset V_G$ be domains of α and β, respectively. The problem is: *Find a path in* G *with minimal summary weight that begins inside* V_α *and ends inside* V_β. This problem is called the *extended shortest path problem (espp)*. So an *espp* means search of a best route that connects two sets, however not arbitrary sets but only those represented by some nodes of the geographic graph T. It was shown in [5], that under some natural assumptions a navigational problem can be solved by a dynamic programming algorithm with a relatively low computational complexity.

2.3 Geography of the Lowest Level

In spite of the theoretical advantages of the GT-model mentioned in the previous section, no progress in its practical implementation can be found in literature since the publication of [5]. The main obstacle for that consists in the necessity of novel methods for automatic learning the lowest, non-verbalizable level of geography. To clarify this issue, consider any geographic item that has an explicit name like Poznan, Broadway, Asia, etc. It is technically clear how to insert such an item into the geographic graph of GT-model. In contrast to that, each item of the lowest geographic level should be represented by a map that does not have neither a distinctive shape, nor an attributed name.

Moreover, in contrast to the verbalizable levels of the geographic graph of the GT-model, the lowest-level geography depends essentially on the sensor system. Note that the higher levels can be the same, say, for a robot, a blind person, as well as for a human who does not suffer any illness of sight. On the contrary, the lowest level geography depends on available sensors: for a blind person – on the stick which he/she uses for exploration of the environment, and for a normal person – on his/her vision. This is why a special technique for learning geography of the lowest level should be developed and studied, and why the detection of heavy changes is important.

2.4 Related Works

A brief review of some related research lines follows.
Simultaneous Localization and Map Building problem (SLAM) [10–13]: the aim of it this approach is to convert a set of robocentric views into a single map. This 'map' is understood not as in our work, but in a traditional, Cartesian meaning so that a human could read it and use. The SLAM techniques involve statistical methods including extended Kalman filter [14] and Rao–Blackwellized particle filters [15]. They allow feeding the map creation while the robot moves smoothly. The ideas of SLAM seem very fruitful to be combined in future with our approach to cope with the robot dynamics.

Path Planning for Autonomous Vacuum Cleaner Robots: The user of a cleaner robot must be certain that in a certain time with a high probability the robot will clean every corner of the workspace. It can be reached avoiding the requirement that the robot any moment be aware of where it is located. So the efficiency of sweeping workspace, sufficient for practice can be provided without construction of a computational model of the workspace. It is reached by applying some context-specific heuristics and sensors in combination with statistical principles [16].

3 A Draft Scheme for Detection of Heavy Changes

3.1 Sensor System for Learning Geography of the Virtual Robot

The main goal of developing the virtual robot briefly presented in the beginning of paper is to study principles of automatic learning non-Cartesian geography.

Image processing and recognition play an auxiliary role. This is why the sensor system of robot was intentionally designed as simple as possible. In particular, each robot view, like one in Fig. 1c, is just a real function of one variable.

It should be mentioned from the beginning that our processing scheme of the robot views depends essentially on the dimension of the views. A discussion about extension in future of this scheme, which is applied here to the 1D-images, to higher dimensions and other kind of sensors, such as conventional video cameras or ultrasonic sensors, lays beyond the scope of this paper.

While using a virtual reality platform it is easy to generate a complex scene and extract any kind of information related to the scene for different kinds of virtual camera. Our virtual robot is developed by means of OpenGL and the screenshots of Figs. 1 are just different ways to render the same scene.

3.2 Main Idea for Detection of Heavy Changes

Intuitively it is clear that the visual events specifically important for robot navigation are related to the facts of appearance/disappearance of objects or gaps in sight and also to a qualitative change in appearance of an object. So the events of such kinds should be primarily detected and then used for indexing the maps of a geographic database. On the other hand, the value of argument where the image of an obstacle begins or ends in a view strongly corresponds to a discontinuity of the first derivative of the plot; then, a discontinuity of the second derivative has a strong correlation with an angle of the object shape. It can be easily seen by comparison of the corresponding elements of Figs. 1b and c connected by the association lines. This observation suggests us to perform a transformation of each angle-distance plot into *string formed by singular points constructed by discontinuities of the first and the second derivatives* of the plot. The transformation 'plot-to-string' is the first step in detection of heavy changes. Its details are presented in the upper block-diagram of Fig. 2.

In this transformation, the type of any singular point belongs to the following short alphabet: {STEP_UP, STEP_DOWN, ANGLE_TOWARD_ROBOT, ANGLE_OUTWARD_ROBOT} with an obvious intuitive meaning of each option.

A subsequent processing step performs temporal analysis of the flow of such strings. The idea to *use for this step a technique based on the Levenshtein distance* suggests itself: when two strings of the singular points are represented in an alphabet, one can find their editorial prescription. And if, for example, such two strings generated for some close moments completely match each other, it is naturally to claim that no heavy change occurred; otherwise, a further analysis of the vector of editorial prescription should follow to classify possible occurrence of a heavy change. A graphic representation of this step with more details is shown in the lower block-diagram of Fig. 2.

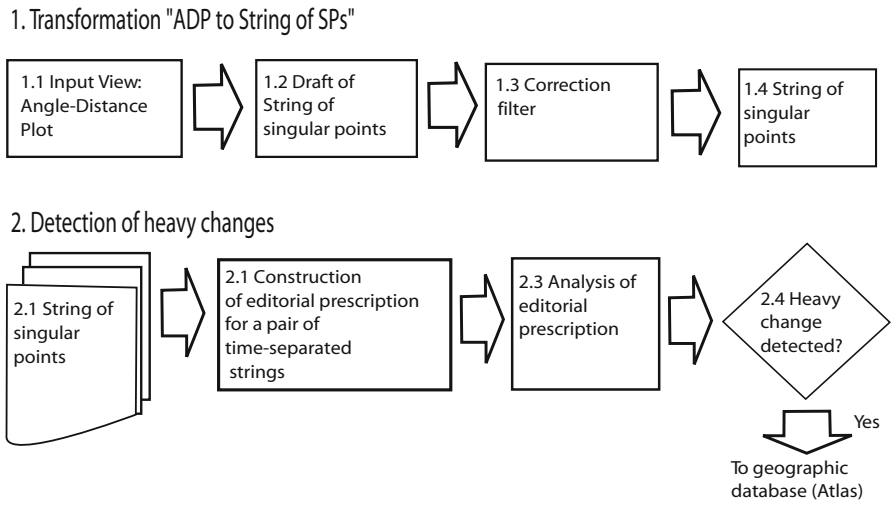

Fig. 2. Block-diagram of the two main steps of detection of heavy changes.

4 Specific Issues of Detection of Heavy Changes

There are two specific issues in the just presented scheme. The former is related to the errors in classifying a singular point, blocks 1.2 and 1.3 of Fig. 2: it turns out to be that such errors cannot be avoided completely. Indeed, if the robot would drive around a pyramid, its shape in the view changes and finally a shape of angle in the angle-distance plot will be transformed to a shape of step-function; consequently the errors are inevitable for some critical region of arguments. Nevertheless, some thresholds of the algorithm of extraction of singular points can be optimized to reduce probability of the errors of this kind. This optimization was provided, [17], and the probability of errors from its initial value 8% for some intuitively chosen thresholds was reduced in result to 0.2–0.3%. Other kind of possible errors is generated by the detection of two or more very close singular points that, in fact, are yielded by a single singular point. The correction filter (Fig. 2, block 1.3) was introduced just to reduce errors of this kind. For example, the filter merges two STEPs of the same kind (see the alphabet in the previous section) if distance between them is very short.

The latter is related to the application of the Levenshtein distance and is a consequence of the fact that as the alphabet for singular points is rather small as well as the strings of singular points are respectively short, which makes the errors in interpretation of editorial prescription (block 2.3 of Fig. 2) rather probable. In the rest of this section, we describe some problem-specific expedients to reduce significantly the probability of the last kind of errors.

4.1 An Enriched Description of Symbols for Computing the Levenshtein Distance

A close look at the process of computation due to formula (1), shows that whatsoever specific of a particular problem is hidden in computation of the function $1_{(a_i \neq b_j)}$ and the costs $w(p, q)$, $w(\varepsilon, q)$, and $w(p, \varepsilon)$. As to function $1_{(a_i \neq b_j)}$ if one just compares literal coincidence of the symbols a_i and b_j then no problem-specific is taken in consideration. However, if each symbol is provided with an enriched problem-specific description, it allows us to improve the proper definition of $1_{(a_i \neq b_j)}$. For the problem under consideration, additionally to its type from the above alphabet, each singular point can be accompanied, for instance, with the value of distance. Since the strings to be compared typically correspond to the two robot states, one before and another after application of a robot control, it is technically possible to evaluate whether the distance associated with b_j can match to the distance of a_i. This idea was implemented in our software, which significantly improved the function $1_{(a_i \neq b_j)}$. See the top-right of Fig. 3 for an instance of computation of $1_{(a_i \neq b_j)}$.

One more expedient to reduce errors consists in the introduction of an artificial symbol of the kind LEG between each pair of usual singular points. Its description, in particular, includes the leg-length, that is, the distance between two successive singular points. This allows us to improve furthermore the idea of Sect. 4.1: using lengths of legs, it is analyzed how probable is that under a certain control two legs can match each other.

4.2 Interpretation of an Editorial Prescription

We construct Editorial Prescription (EP) by the well-known Wagner-Fischer algorithm [18]. Finally, an EP is represented as a string in the following alphabet: {MATCH, REPLACE, INSERT, and DELETE }. Evidently, if all symbols of an EP are MATCHes, then no heavy change occurs. If EP contains INSERT or DELETE, it mostly means a heavy change. But some situations regarded in Block 2.4 of Fig. 2 require a problem-specific heuristic. Just two examples of such heuristics for computation of $1_{(a_i \neq b_j)}$ follow:

A limited equivalence between MATCH and REPLACE is allowed. For example, an STEP_UP is regarded as equivalent to ANGLE_TOWARD_ROBOT for singular points with distances quite close to the far distance-limit of angle-distance plots. This decision naturally follows from the manner of how OpenGL constructs the content of z-buffer [19]: when some real distances are far, they are compressed in z-buffer to very close values; so it is easy to confuse a far STEP with a far ANGLE if they are of appropriate types. Partly, interpretation of an EP depends on a priori knowledge of the applied robot control. For example, if the control command is ROTATE, it is expected that distance to any singular points will stay practically the same after application of the command. A rotation can lead to an appearance or disappearance of a singular point at the periphery of the field of view. If this occurs, we assume that a disappearance of singular point is not a heavy change, but an appearance is a heavy change.

The above heuristics related to the indicator function were combined with a problem-specific choice of the costs in formula (1). It was found experimentally that for our case they should not all be the same. We use the cost of replacement $w(p, q)$ two times bigger than the costs of deletion $w(\varepsilon, q)$ and insertion $w(p, \varepsilon)$, which are set as equal.

5 Experiments

All experiments were accomplished using a pilot software mentioned above. A detailed description of the software lay beyond the scope of this paper. Just a concise list of its functional components follows:

- Interactive creation of obstacles of the robot environment (based on [20]);
- Visualization of the environment and different kinds of robot views (based on [20]);
- Intuitive supervisor graphic interface for controlling robot;
- Training associative memory (Kohonen) to be able to reconstruct a robot control that drives the robot from one view to another;
- Extraction of singular points and transformation of angle-distance plots to a string (see Sect. 3.1 and beginning of Sect. 4);
- Construction of Editorial Prescription (EP) for two strings (see Sects. 2–4);
- Detection of a heavy change by interpretation of a EP (see Sects. 3, 4);
- Support of the self-localization task;
- Manipulations with the atlas of geographic maps;
- Serialization/Deserialization.

Using software comprises three stages: (i) generation of obstacles inside the workspace; (ii) training the associative memory; (iii) learning geography of the workspace. Some details of this process follow.

Supervisor puts on the scene any number of obstacles (pyramids or cubes), controlling their size, shape, position, and orientation. Then he/she drives the robot randomly around the scene; this leads to automatic training of matrices of the Kohonen associative memory (the data generated at this stage are needed for automatic estimation of a robot control command that would drive the robot from one view to some other close view). When the memory is trained, the program automatically initializes the robot position as in Fig. 1a and sets all content of the geographic atlas to a single initial map.

Starting from this point, supervisor generates discrete commands to drive randomly the robot around the scene by means of an intuitive graphic interface. The robot executes the command sequence and automatically learns geography of the workspace: each robot control introduced by supervisor invokes the above scheme of detection of heavy changes; any detected heavy change means that the current map must be changed; the new current map is chosen from the neighbors of the old one or, if it is impossible, a new map is introduced into the atlas and then assigned as the current map.

Fig. 3. A short sequence of screen-shots of a learning geography experiment (see the main text).

The correctness of the principles of detection of heavy changes presented in this work should follow from the fact that the robot would demonstrate the ability to learn correctly the geography. The *criterion of correctness* of all taken decisions is as follows: when a robot trajectory returns to a past point with the same orientation, its current map must return to an old map that robot had as the current while visiting this point the last time.

A learning geography experiment is illustrated in Fig. 3. The beginning of the learning process corresponds to the frames of Fig. 1. Then the robot was ordered to perform three turns without any change of position: two successive counter-clockwise turns and a reverse turn with the same angle. The three angle-distance plots show the dynamics of the robot views. Note that each of the three screens shows simultaneously a previous and actual views, respectively, as a stippled and continuous line-strip. Three screenshots of the console window, from the top-right to the bottom-left, correspond to respective robot views and show the response of the learning algorithm; each of them contains an editorial prescription, a result of detection of heavy change, if applicable, and a subsequent

action with geographic database. Additionally, the first console-window contains an example of computation of indicator function $1_{(a_i \neq b_j)}$.

It can be seen that no heavy change occurs after the first turn and so robot stays in the same map, ID 0; then a heavy change occurs and robot introduces a new map, ID 1, into the atlas, and changes the actual map to the new one; after the reverse turn, robot again detects heavy change and returns to map ID 0 chosen as a neighbor of map ID 1.

The results of this simplest experiment completely satisfy the criterion mentioned above. A series of such experiments was implemented, including those with a much longer sequence of robot commands. The experiments which did not pass the above criterion were used for tuning components of the package; especially, the tuning is concerned with the issues described in Sect. 4. After the tuning, all succeeding experiments accomplished so far under normal conditions satisfy the criterion. The 'normal conditions' mean that any view contains a sufficient number of singular points and that the possibility of two similar views for distant robot states on a robot path is excluded. The dashed line in Fig. 1a shows a typical example of a long robot trajectory fulfilled in an experiment after the tuning. The whole route up to closing the first cycle corresponds to 60 commands of supervisor. The robot detects correctly the cycles of the path and interacts with the atlas in full correspondence with the expectations.

6 Conclusions

This work is oriented to application of the non-Cartesian navigation in robotics. Presented results for the first time show how automatic learning the geography of a robot environment by means of processing the flow of robot views can be implemented. The presented here approach is based on detection of heavy changes in the input flow and their use for interaction with a geographic database. In its turn, the detection of heavy changes is provided by transformation of each view to a string with subsequent comparison of such strings by a technique based on application of the Levenshtein distance. This scheme is implemented in the scope of a pilot software that simulates actions of a virtual robot in 2D-environment and integrates all components of the approach. The first series of experiments with this software allows us to claim that main principles presented here are correct and lead to automatic learning the geography.

Main research lines for future work are: replacement of the 1D robot sensor to other kinds of sensors; extension of the technique from the 1D to 2D views; extension to a non-static environment; the connection of this approach with the possibility of setting navigation problems in natural terms, etc. As to some challenging goals oriented to practice, we can mention creation of a drone that would be able to navigate automatically in 3D workspace, and connecting our approach with techniques based on GPS.

Acknowledgment. UAM-Azcapotzalco supports this work in the scope of project "Artificial Cognitive Vision".

References

1. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Dokl. Akad. Nauk SSSR **163**, 845–848 (1965)
2. Su, Z., Ahn, B.R., Eom, K.Y., Kang, M.K., Kim, J.P., Kim, M.K.: Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm. In: Innovative Computing Information and Control, ICICIC 2008, pp. 569–569. IEEE Press (2008)
3. Heeringa, W.J.: Measuring dialect pronunciation differences using Levenshtein distance. Doctoral dissertation. University Library Groningen (2004)
4. Schimke, S., Vielhauer, C., Dittmann, J.: Using adapted levenshtein distance for on-line signature authentication. In: Pattern Recognition, 2004 ICPR 2004, pp. 931–934. IEEE Press (2004)
5. Khachaturov, G.: An approach to trip-and route-planning problems. Cybern. Syst. **33**, 43–67 (2002)
6. Gomi, T.: Non-cartesian robotics. Robot. Autonom. Syst. **18**, 169–184 (1996)
7. Gomi, T.: Aspects of non-cartesian robotics. Artif. Life Robot. **1**, 95–103 (1997)
8. Vukobratović, M.: How to control robots interacting with dynamic environment. J. Intell. Rob. Syst. **19**, 119–152 (1997)
9. van Leeuwen, J.: Graph algorithms. In: Handbook of Theoretical Computer Science: Algorithms and Complexity, vol. A, pp. 525–631 (1990)
10. Smith, R., Self, M., Cheeseman, P.: A stochastic map for uncertain spatial relationships. In: Robotics Research: The Fourth International Symposium, pp. 467–474 (1988)
11. Dissanayake, M.G., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Autom. **17**, 229–241 (2001)
12. Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 865–880 (2002)
13. Guivant, J., Nebot, E., Baiker, S.: Autonomous navigation and map building using laser range sensors in outdoor applications. J. Robotic Syst. **17**, 565–583 (2000)
14. Castellanos, J.A., Martinez-Cantin, R., Tardós, J.D., Neira, J.: Robocentric map joining: improving the consistency of EKF-SLAM. Robot. Autonom. Syst. **55**, 21–29 (2007)
15. Grisetti, G., Tipaldi, G.D., Stachniss, C., Burgard, W., Nardi, D.: Fast and accurate SLAM with Rao-Blackwellized particle filters. Robot. Autonom. Syst. **55**, 30–38 (2006, 2007)
16. Hasan, K.M, Reza, K.J., Abdullah-Al-Nahid.: Path planning algorithm development for autonomous vacuum cleaner robots. In: Informatics, Electronics Vision 2014, pp. 1–6. IEEE Press (2014). doi:10.1109/ICIEV.2014.6850799
17. Khachaturov, G., Espinosa de los Monteros, J.A., Figueroa, J.: Applying spatio-temporal analysis to angle-distance views for detection of relevant events. In: Information Technology: Proceedings of Conference on Informatics and Computer Science (CNCIIC-ANIEI 2016), Mexico, pp. 205–214 (2016)
18. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**, 31–88 (2001). doi:10.1145/375360.375365
19. Joy, K.: The depth-buffer visible surface algorithm. Visualization and Graphics Research Group, Department of Computer Science, University of California
20. Martínez-Hernández, J.M.: Development of a virtual robot and its environment based on a non-Cartesian navigation model. Terminal project for graduation in Computer Engineering from UAM-Azcapotzalco (2010) (in Spanish)